

Amendments to the Claims

1 Claim 1 (currently amended): A method for programmatically enforcing referential integrity
2 constraints among associations between class instances, comprising steps of:
3 programmatically determining, when evaluating a request to set an association end to
4 reflect an association from an instance of a first class to an instance of a second class, whether
5 the association end to be set has a single multiplicity or a many multiplicity;
6 if the association end to be set has the single multiplicity, atomically and
7 programmatically performing the steps of:
8 setting the requested association end; and
9 programmatically modifying
10 setting an inverse association end of the association to reflect an inverse
11 association from the instance of the second class to the instance of the first class, after
12 disconnecting the inverse association end from an existing instance of the second class, if any;
13 and
14 setting the requested association end from the instance of the first class to the
15 instance of the second class; and
16 if the association end to be set has the many multiplicity, atomically and
17 programmatically performing the steps of:
18 adding the requested association end to the instance of the first class; and
19 setting an inverse association end of the association to reflect an inverse
20 association from the instance of the second class to the instance of the first class, after
21 disconnecting the inverse association end from an existing instance of the first class, if any.

22 wherein an ordering of the setting step and the programmatically modifying step depends
23 on an outcome of the determining step.

Claims 2 - 4 (canceled)

1 Claim 5 (currently amended): The method according to Claim 1, further comprising [[steps]] the
2 step of[:]] serializing the association by performing steps of:
3 determining whether the association end to be set or the inverse association end is a
4 primary end of the association; and
5 serializing only the primary end of the association during [[a]] the serialization operation.

1 Claim 6 (currently amended): The method according to Claim 1, wherein the method is provided
2 as a single link helper objects object and a multiple link helper object for each association,
3 wherein the single link helper object performs the atomically and programmatically performed
4 steps for the single multiplicity association end and the multiple link helper object performs the
5 atomically and programmatically performed steps for the many multiplicity association end.

1 Claim 7 (currently amended): A computer program product for programmatically enforcing
2 referential integrity constraints among associations between class instances, wherein the
3 computer program product is embodied on one or more computer readable media and comprises
4 computer-readable program code means for:
5 computer-readable program code means for programmatically determining, when

6 evaluating a request to set an association end to reflect an association from an instance of a first
7 class to an instance of a second class, whether the association end to be set has a single
8 multiplicity or a many multiplicity;

9 computer-readable program code means for setting the requested association end; and
10 if the association end to be set has the single multiplicity, atomically and
11 programmatically performing the steps of:

12 — computer-readable program code means for programmatically modifying
13 setting an inverse association end of the association to reflect an inverse
14 association from the instance of the second class to the instance of the first class, after
15 disconnecting the inverse association end from an existing instance of the second class, if any;
16 and

17 setting the requested association end from the instance of the first class to the
18 instance of the second class; and

19 if the association end to be modified has the many multiplicity, atomically and
20 programmatically performing the steps of:

21 adding the requested association end to the instance of the first class;
22 setting an inverse association end of the association to reflect an inverse
23 association from the instance of the second class to the instance of the first class, after
24 disconnecting the inverse association end from an existing instance of the first class, if any.

25 — wherein an ordering of operating the computer-readable program code means for setting
26 and the computer-readable program code means for programmatically modifying depends on an
27 outcome of the computer-readable program code means for determining.

Claims 8 - 9 (canceled)

1 Claim 10 (currently amended): The computer program product according to Claim 7, further
2 comprising computer-readable program code means for serializing the association by performing
3 steps of:
4 computer-readable program code means for determining whether the association end to
5 be set or the inverse association end is a primary end of the association; and
6 computer-readable program code means for serializing only the primary end of the
7 association during [[a]] the serialization operation.

1 Claim 11 (currently amended): A system for programmatically enforcing referential integrity
2 constraints among associations between class instances, comprising means for:
3 programmatically means for determining, when evaluating a request to set an association
4 end to reflect an association from an instance of a first class to an instance of a second class,
5 whether the association end to be set has a single multiplicity or a many multiplicity;
6 means for setting the requested association end; and
7 if the association end to be modified has the single multiplicity, atomically and
8 programmatically performing the steps of:
9 means for programmatically modifying
10 setting an inverse association end of the association to reflect an inverse
11 association from the instance of the second class to the instance of the first class, after

12 disconnecting the inverse association end from an existing instance of the second class, if any;
13 and

14 setting the requested association end from the instance of the first class to the
15 instance of the second class; and

16 if the association end to be modified has the many multiplicity, atomically and
17 programmatically performing the steps of:

18 adding the requested association end to the instance of the first class; and

19 setting an inverse association end of the association to reflect an inverse
20 association from the instance of the second class to the instance of the first class, after
21 disconnecting the inverse association end from an existing instance of the first class, if any.

22 ————— wherein an ordering of operating the means for setting and the means for
23 programmatically modifying depends on an outcome of the means for determining.

Claims 12 - 13 (canceled)

1 Claim 14 (currently amended): The system according to Claim 11, further comprising means for
2 serializing the association by performing steps of:

3 means for determining whether the association end to be set or the inverse association end
4 is a primary end of the association; and

5 means for serializing only the primary end of the association during [[a]] the serialization
6 operation.

1 **Claim 15 (new):** The method according to Claim 1, wherein one or more structured markup
2 language representations specify instances of the first class, instances of the second class, and
3 associations between the instances of the first and second classes.

1 **Claim 16 (new):** The method according to Claim 15, wherein only one association end for each
2 association between instances is specified in the structured markup language representations.

1 **Claim 17 (new):** The method according to Claim 16, wherein the only one association end is an
2 association end designated as a primary end for the association.

1 **Claim 18 (new):** The method according to Claim 15, wherein a serialization of results of the
2 request to set the association end that has the single multiplicity comprises the step of:
3 determining whether the association end to be modified is a primary end for the
4 association, and if so, programmatically performing the steps of:

5 removing the representation of the previously-existing inverse association end, if
6 any, from the structured markup language representation in which it is specified; and
7 adding a structured markup language representation of the new inverse association
8 end.